

MAPPING RULES FOR ER TO XML USING XML SCHEMA

Sung Jin

Colorado State University Pueblo

s.jin@colostate-pueblo.edu

Woohyun Kang

Colorado State University Pueblo

w.kang@colostate-pueblo.edu

Abstract

The Extensible Markup Language (XML) has emerged as a de facto standard language for exchanging data among disparate information systems. Since the majority of data in the world is stored in databases, the conversion of such data into XML documents is indispensable for real world usage. In this paper, we propose the algorithm for mapping the Entity Relationship model (ER model) to the XML document at the metadata level to address this conversion issue effectively. This is a progressive attempt to automatically generate the XML document from the conceptual database schema using the XML Schema. In this paper, we introduce a set of ER-to-XML mapping rules and then show the mapping algorithm based on these rules. We also introduce the XML Schema which forms the groundwork of this mapping algorithm

Keywords: ER, XML Schema, Mapping Algorithm

Introduction

Motivation

Since the Extensible Markup Language (XML) (W3C, 2004a) was developed by World Wide Web Consortium (W3C) in 1996, it has been adopted in numerous fields for various purposes. Now the XML is considered as a de facto standard language for exchanging data among disparate information systems. The Document Type Definition (DTD) has been used to describe XML data structures and constraints. However, due to its limitations such as data typing, content modeling, and extensibility, several alternatives for describing the XML structure are proposed. They include RELAX (Murata, 2001), TREX (Clark, 2001), Relax NG (Clark & Murata, 2001), DSD (Moller, 2003), and XML Schema (W3C, 2004b). Among these, XML Schema of W3C is regarded as the most acceptable one, because it is recommended as a standard by W3C and its development is supported by almost all major IT companies.

Since the majority of data in the world is stored in databases, the conversion of such data into XML documents is indispensable for real world usage. In this conversion, rules and algorithms for preserving the information of the database schema and generating XML documents based on such information are necessary.

In previous research, rules and algorithms to map data between XML and databases have been proposed (Carey et al, 2000, Fernandez et al, 2000, Florescu & Kossmann, 1999, Kleiner & Lipeck, 2001, Lee et al, 2003, Shanmugasundaram et al, 1999, and Turau, 1999). Most researchers so far, however, use DTD instead of XML Schema in their research, so their results are often insufficient or inefficient. Moreover, careful intervention of a human expert is indispensable. To overcome this deficiency, we propose ER-to-XML mapping rules at the schema level using the XML Schema. By using the XML Schema, we can express mapping rules more efficiently and address the limitations of DTD. We also present an algorithm, which automatically generates XML-based ER schemas.

The remainder of this paper is organized as follows. In section 2, after a brief re-view of the related research, we present the ER-to-XML mapping rules. In section 3, we show the algorithm based on the ER-to-XML mapping rules. Some concluding remarks are presented in section 4.

Related Work

In the last few years, XML-to-Database mapping issues have been studied by several researchers. Shanmugasundaram et al. (1999) develop algorithms that convert XML documents to relational tuples, and translate semi-structured

queries over XML documents to SQL queries over tables. Florescu & Kossmann (1999) conduct comparative analysis of eight algorithms which store the XML data in relational database systems. Lee et al. (2003) propose three semantics-based schema conversion methods between DTD and relational schema.

In the study of Turau on the transformation of data from the relational database into the XML document, the relational database schema is generated in form of the DTD (Turau, 1999). Kleiner & Lipeck (2001) present an algorithm that can generate DTDs from conceptual database schemas. Fernandez & Suci (2000) develop a query language that can transform relational data to XML data. Carey et al. (2000) publish the XML data from the object-relational database.

Mapping Rules

In this section, we describe ER-to-XML mapping rules at the schema level. The procedures of each rule are illustrated using a simple example. The mapping rules are concise and easy to understand. Each entity type and relationship type in the ER diagram is mapped into the top-level element in the XML document. There are 6 top-level XML elements that represent different entity types and relationship cardinalities: `<entity>`, `<weak-entity>`, `<unary-relationship>`, `<binary-relationship>`, `<ternary-relationship>`, and `<n-ary-relationship>`.

The content (i.e., data value) of a top-level element is the same as the corresponding name of an entity type or a relationship type. For example, an entity type `STUDENT` is represented in XML as `<entity>STUDENT</entity>`. The attributes of an entity type in the ER diagram are mapped into the sub-element `<attribute>` of the corresponding top-level element in XML (i.e., `<entity>` or `<weak-entity>`). Similarly, the attributes and other meta-information of a relationship type in the ER diagram (e.g., cardinality constraints, role names of the relationship, participating entities, etc.) are described at the sub-element of the corresponding element (i.e., `<unary-relationship>`, `<binary-relationship>`, etc.). The detailed mapping rules are explained in the following subsections.

Entity

The strong entity type S in the ER diagram is mapped to the `<entity>` element in the XML document as shown in Figure 1. The mapping rule of the weak entity type is similar to that of the strong entity type. The weak entity type W in the ER diagram is mapped to the `<weak-entity>` element. Figure 1 presents the weak entity to the XML element mapping example.

	ER	XML
Strong Entity	<div style="border: 1px solid black; padding: 5px; display: inline-block;">MOVIE</div>	<code><entity>MOVIE</entity></code>
Weak Entity	<div style="border: 3px double black; padding: 5px; display: inline-block;">DEPENDENT</div>	<code><weak-entity> DEPENDENT </weak-entity></code>

Figure 1. Entity

Attribute

Simple attribute

A simple attribute A of the entity E in the ER diagram is represented in XML using the `<attribute>` element. The `<attribute>` element is placed as a sub-element of the belonging top-level XML element.

Note that other attribute mapping rules described in the ensuing sections (i.e., key attribute, composite key attribute, multi-valued attribute, and composite attribute) are based on this simple attribute mapping rule. This property allows us to define the structure of the simple attribute as an XML complex type called attribute-type and reuse it multiple times in the XML Schema. This example is presented later in section 2.2.5. Figure 2 shows the mapping relationship between the attribute `Profile` of the entity type `ACTOR` and its corresponding XML document and XML schema. Note that in XML schema, the XML attribute data-type in the XML complex type attribute-type is defined optional. The reason is that in most ER diagrams the data type of attributes is not specified.

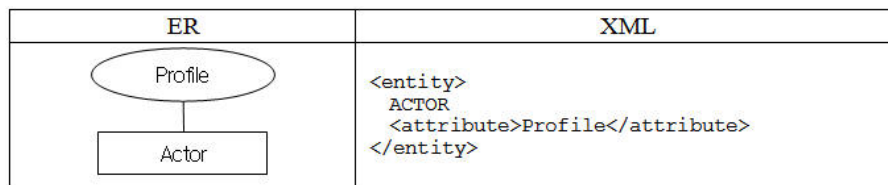


Figure 2. Simple Attribute

Key Attribute

The key attribute A of the entity E is mapped in a similar way to the simple attribute. In this case, the `<key-attribute>` element is added as a sub-element of the top-level element E as shown in Figure 3.

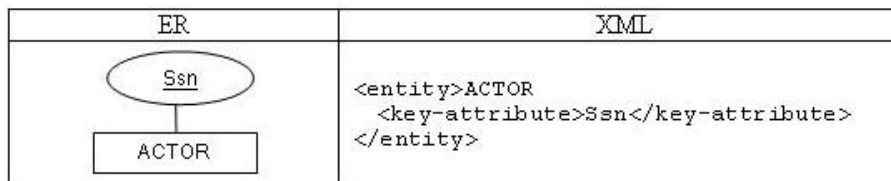


Figure 3. Key Attribute

If the key attribute consists of A_1, A_2, \dots, A_n of the entity type S (i.e., composite key attribute), the `<composite-key-attribute>` element is used as the parent element of a set of `<key-attribute>`.

Multi-valued Attribute

In some cases, an attribute can have a set of values for the same entity (i.e., multi-valued attribute). The multi-valued attribute in the ER diagram is mapped to the `<multivalued-attribute>` element. The name of the multi-valued attribute A of the entity E is mapped to the value of the `<multivalued-attribute>` element as illustrated in Figure 4.

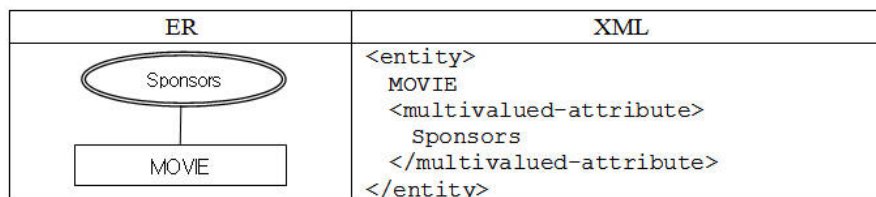


Figure 4. Multi-valued Attribute

Derived Attribute

When two or more attribute values are related, the value of one attribute can be determined by the other attribute(s) (i.e., derived attribute). The derived attribute in the ER diagram is represented as the `<derived-attribute>` element in the XML document as shown in Figure 5.

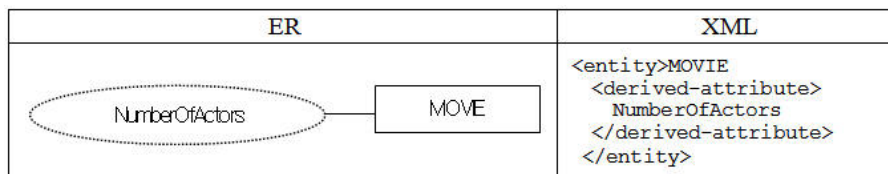


Figure 5. Derived Attribute

Composite Attribute

If the attribute consists of A_1, A_2, \dots, A_n of the entity type S (i.e., composite attribute), the `<composite-attribute>` element is used as the sub-element of the top-level element S . A set of `<attribute>` is placed as a sub-element of the `<composite-attribute>` element as shown in Figure 6. All three attributes, $Fname$, $Minit$, and $Lname$ in Figure 6, are simple attributes and described using the `<attribute>` elements. As discussed previously, we declare the simple attribute as the XML complex type called attribute-type. This named type feature of the XML Schema enforces consistency, reduces errors, and simplifies maintenance.

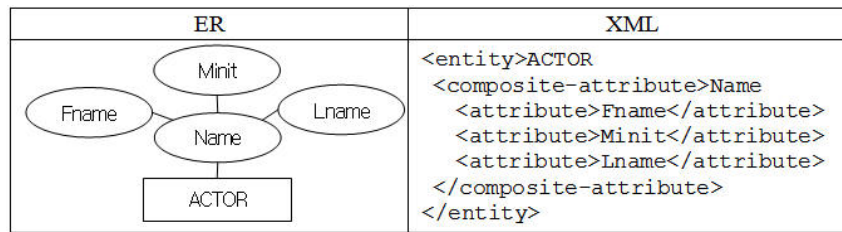


Figure 6. Composite Attribute

Relationships

Binary Relationship

The binary relationship R between two entity types S and T is mapped to the top-level element `<binary-relationship>`. If the relationship type has attributes, each attribute is also mapped to the sub-element `<attribute>` of the matched top-level `<binary-relationship>` element. In addition, the two participating `<entity>` elements are also placed as sub-elements. In this case, for the associated `<entity>` element, there are two required XML attributes to express the minimum and maximum cardinality constraints (i.e., `min-card` and `max-card`, respectively) and one optional XML attribute for the role name in the ER diagram (i.e., `role-name`). It is worth noting that this participating `<entity>` element is defined by `participating-entity-type` in the XML schema which is derived from the XML complex type entity-type (see Figure 7).

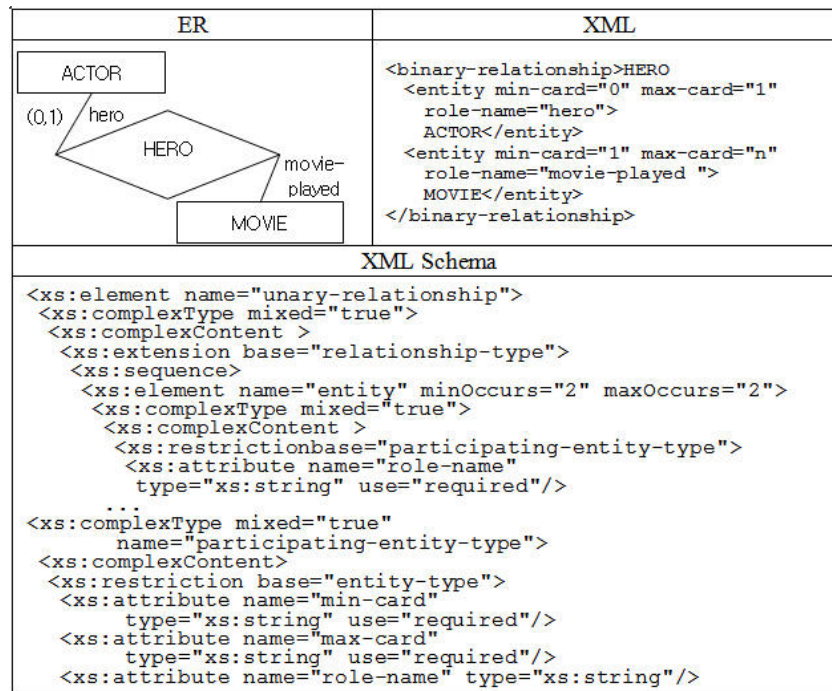


Figure 7. Binary Relationship

Unary Relationship

In some cases, a relationship consists of only one entity type. Thus, the two `<entity>` elements which are placed as the sub-elements of matched top-level `<unary-relationship>` element should have the same entity type. In this case, XML attribute `role-name` is required instead of optional as shown in Figure 8. The reason for enforcing the strict rule is that if the role name is not specified in the unary relationship, the semantics of the relationship are vague.

n-ary Relationship

If the relationship consists of more than three different entity types, each entity type is mapped to the sub-element `<entity>` of the corresponding top-level `<n-ary-relationship>` element. In this case, our XML Schema rule enforces that the total number of `<entity>` elements should be at least four.

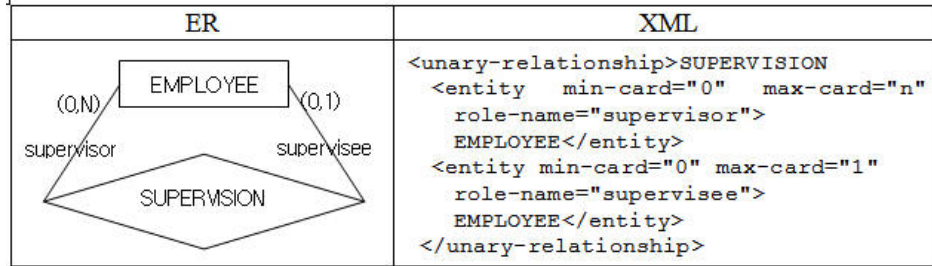


Figure 8. Unary Relationship

Mapping Algorithm

In this section, we show the mapping algorithm based on the ER-to-XML mapping rules. The XML Schema is used as an input with the ER diagram in our algorithm to validate the structure of the XML document. The complete algorithm in pseudo-code is presented in Figure 9.

By following *Step 1* through *Step 5* based on the ER-to-XML mapping rules in an appropriate manner, we can represent all the information in the ER diagram as an XML document correctly. For a clearer explanation, we illustrate the ER-to-XML algorithm step by step using simple ER diagram (See. http://student.colostate-pueblo.edu/w.kang/simple_ER-diagram.jpg).

Algorithm: ER-to-XML Mapping

Input: ER diagram, and Predefined mapping-purposed XML Schema

Output: XML document

Method:

- Step 1:* Define the root element. First, we generate a root element of the XML document. The name of the root element is `<ER-Model>`. The content of the root element is determined by the name of the ER diagram.
- Step 2:* Mapping entity types. First, we identify the entity types in the ER diagram including the weak entity type. Second, we map the entity types according to the entity type mapping rules. All the XML elements generated in this step (i.e., `<entity>` and `<weak-entity>`) must be placed at the top-level.
- Step 3:* Mapping attribute types. After mapping all the existing entity types, attributes of each entity types are checked in step 3. The key attribute type mapping must be checked first among the other attribute types. After mapping the key attribute types, all the other attribute types are mapped one after another in accordance with the attribute mapping rules. We can choose the mapping order of attribute type at our convenience until all the attributes of each entity types are used up.
- Step 4:* Mapping relationship types. After mapping the entity related information in the ER diagram, we identify the relationship types in ER diagram. Each relationship type is mapped to the matched XML element (i.e., `<unary-relationship>`, `<binary-relationship>`, `<ternary-relationship>` and `<n-ary-relationship>`) at the top-level in random order. Then, we add the content of a top-level element that is same as the corresponding name of a relationship type.
- Step 5:* Mapping semantics of each relationship type. After identifying all the relationship types, we map the attribute types and all the other meta-information of the relationship types in this step. First, we map the attribute types. Then, all the other meta-information of the relationship types is mapped as the sub-element `<entity>` and its XML attributes.

Figure 9. ER-to-XML Algorithm

For *Step 1*, we define and generate the root element of our XML document which is the `<ER-Model>`. After generating the root element, we identify the entity types in the ER diagram. In this case, two strong entities (e.g., `<ACTOR>` and `<MOVIE>`) and one weak entity (e.g., `<DEPENDENT>`) are identified. *Step 3* is assigning all the attributes to the proper elements. Thirteen attributes are identified from our example. In this step, it is important that key attribute type mapping be performed prior to the other attribute types such as simple attribute, composite attribute, etc. The order of the other attribute type mappings is optional. After mapping all six attributes of the `<ACTOR>` element, we perform the same procedure for each element. The fourth step is identifying the relationship

types between entities. As we designated, there are four relationship types: unary, binary, ternary, and n-ary. For example, the HERO relationship, which explains the relationship between <ACTOR> and <MOVIE>, has only two participant entities in its relationship. So we identify HERO as a binary-relationship. After identifying the relation type, we identify the meta-information such as information about the cardinality. <ACTOR> element can play a hero role in this relationship. In most case, however, not every actor plays a hero in one movie. That is the reason why this element has cardinality (0, 1). After identifying all the relationship types and meta-information, we establish the relationship between every entity as we illustrated in section 2.3. We assign the HERO relationship to the binary-relationship type. For the hero role, we designated min-card=0 and max-card=1. Similarly, movie-played role is granted min-card=1 and max-card=n cardinality.

This mapping example proves the simplicity and efficiency of our ER-to-XML mapping algorithm. Because we already constructed the predefined mapping-purpose XML Schema, not only experts but also novices in data modeling can accomplish the ER-to-XML mapping with any ER diagram easily without any mistake.

Conclusion

In this paper, we show the ER-to-XML mapping rules for each component in the ER model and present an algorithm for mapping the ER model to the XML document at the schema level based on these ER-to-XML mapping rules. We also introduce the XML Schema which forms the groundwork of this mapping algorithm. Our algorithm is concise but powerful enough to express ER models in XML form without losing any semantics. Our current work includes the implementation of automatic translation tool based on the design view. A complete XML document example used in this paper and the XML Schema that specifies these rules can be downloaded at: <http://student.colostate-pueblo.edu/w.kang/er2xml.html>.

References

- Carey, M., Florescu, D., Ives, Z., Lu, Y., Shanmugasundaram, J., Shekita, E. & Subramanian, S. (2000), XPERANTO: Publishing Object-Relational Data as XML, Proc. International Workshop on the Web and Databases, USA, 2000, 105-110
- Clark, J. (2001), TREX - Tree Regular Expressions for XML: language specification, Web Page, 2001, <http://www.thaiopensource.com/trex/spec.html>
- Clark, J. & Murata, M. (2001), RELAX NG Specification, 2001, <http://www.oasis-open.org/committees/relax-ng/spec-20011203.html>
- Fernandez, F., Tan, C. & Suciu, D. (2000), SilkRoute, Trading between Relations and XML, Proc. 9th International World Wide Web Conference, Netherlands, 2000, 723-745
- Florescu, D. & Kossmann, D. (1999), Storing and Querying XML Data Using an RDBMS, *IEEE Data Engineering Bulletin* (22), 1999, 27-34
- Kleiner, C. & Lipeck, U. (2001), Automatic Generation of XML DTDs from Conceptual Database Schemas, Proc. Workshop of the Annual Conference of the German and Austrian Computer Societies, Austria, 2001, 396-405
- Lee, D., Mani, M. & Chu, W. (2003), Schema Conversion Methods Between XML and Relational Models, *Knowledge Transformation for the Semantic Web*, IOS Press, Netherlands, 2003, 1-17
- Murata, M. (2001), Document description and processing languages - regular language description for XML (RELAX): Part 1: RELAX core, Technical report, ISO/IEC, 2001
- Moller, A. (2003), Document Structure Description 2.0, 2003, <http://www.brics.dk/DSD/dsd2.html>
- Shanmugasundaram, J., Tufte, K., He, G., Zhang, C., DeWitt, D. & Naughton, J. (1999), Relational Databases for Querying XML Documents: Limitations and Opportunities, Proc. International Conference on Very Large Data Bases (VLDB), Scotland, 1999, 302-314
- Turau, V. (1999), Making Legacy Data Accessible for XML Applications, 1999, <http://www-1.informatik.fh-wiesbaden.de/~turau/DB2XML/index.html>
- W3C (2004a), Extensible Markup Language (XML) 1.0 (Third Edition), World Wide Web Consortium, 2004, <http://www.w3.org/TR/2004/REC-xml-20040204/>
- W3C (2004b), XML Schema Part 0: Primer, World Wide Web Consortium, 2004, <http://www.w3.org/TR/xmlschema-0/>